AN ACCURATE INDUCTION METHOD OF PLAYER RATINGS FROM TOURNAMENT RESULTS

Frederic Maire and Rune Rasmussen

School of Software Engineering and Data Communication,
IT Faculty, Queensland University of Technology,
2 George Street, GPO Box 2434,
Brisbane Q 4001, Australia.
f.maire@qut.edu.au , r.rasmussen@student.qut.edu.au

ABSTRACT

Competitive co-evolution has been successfully applied to breed artificial players. The fitness of players is evaluated through competition with other players. The estimation of the fitness (rating) of players is the computational bottleneck of this approach. It is therefore desirable to exploit as best as possible the information contained in the outcomes of played games. In this paper, we introduce an efficient induction method of player ratings from tournament results. We demonstrate empirically that the proposed formula gives more accurate results than the estimation formula that is traditionally used.

Keywords: player rating, fitness function, tournament, genetic algorithm.

1. INTRODUCTION

In competitive co-evolution [11], the fitness of individuals is evaluated through competition with other individuals in the population, rather than through an absolute fitness measure. In this context, fitness signifies only the relative strengths of solutions; an increased fitness in one solution leads to a decreased fitness for another. This process can be used to evolve solutions that are difficult to evolve in terms of an absolute fitness function. Ideally, co-evolution leads to an arms race of increasingly better solutions within the population.

Competitive co-evolution has been successfully applied to a wide range of games, including Simulated Hockey [3], Checkers [4], Backgammon [8], Poker [1], Chess [14, 2] and even Go [7].

All evolutionary approaches require the evaluation of the fitness of the individuals in the population. The following protocol is generally adopted for evolving artificial players [4]. At each generation, players in the current population participate in a tournament (any competition for players in which a series of games is played). The performance of a player in a tournament determines the likelihood of this player being selected for the creation of the next generation (likelihood of surviving to the next generation and mating with other players).

Running a tournament is extremely expensive computationally, as completing a single game may requires several minutes. In order to best exploit the information contained in tournament results, it is desirable to obtain an accurate estimate of the relative strength of the players based on tournament results. The relative strength of players can be quantified with a rating system. For humans, game federations maintain player ratings with an adaptive formula. However, these adaptive rating systems are not well suited for genetic algorithms as no prior history of the players is available in a typical evolutionary context. Therefore, it is desirable to have an efficient induction method to accurately estimate the true ratings of artificial players from a single tournament.

In this paper, we introduce an efficient induction method of player ratings from tournament results. We demonstrate empirically that the formula proposed gives more accurate results than the estimation formula that is traditionally used.

In Section 2, we review previous work on evolving artificial players for strategic games. In Section 3, we briefly describe the ELO rating system. In Section 4, we introduce a method to directly derive player ratings from a tournament. In Section 7, we present experimental results.

2. EVOLVING ARTICIAL PLAYERS

Since the beginning of the twentieth century, computing scientists have tried to model strategic games to create expert artificial players. Chess has been one of the most researched of these games [6]. After several decades of research focusing on the creation of grandmaster standard computer programs, this collective effort research culminated in the

defeat of Garry Kasparov, the World Chess Champion, by IBMs purpose-built chess computer called Deep Blue, in 1997.

To go beyond brute force methods, artificial players need better evaluation functions for predicting the outcome of games. The automated learning of evaluation functions is a promising research area of genetic programming. Around 1949, Shannon pionneered the idea of evaluation function for chess programs [10]. A decade later, Samuel [9] developed a checkers program that employs learning by using the outcome of games between two players. In the late 1980's, Sutton [12] developed Samuels ideas further in the framework of reinforcement learning by introducing methods for Temporal Difference Learning (TDL). Many researchers have since applied TDL to a number of strategic games, including chess [14, 2] and Backgammon [13].

Co-evolutionary methods pair artificial players in competitions and selection is used to eliminate those that perform poorly relative to other players. In 1999, Chellapilla and Fogel successfully developed strategies for playing checkers with the use of a population of neural network candidate players [4]. The process of evolution only requires the final aggregated outcome of each game played (i.e., win, lose, or draw). Barone [1] applied adaptive learning to produce a good poker player. It competed in a worldwide tournament involving several human expert players and achieved a ranking of top 22. Another successful example of co-evolution is Pollacks Backgammon player [8].

In all the experiments reported in the literature, the fitness of an evolved player is assessed by simply counting the number of wins and losses of the player. We called this assessment method *the standard estimator*. With this method, the strength of the opponent is not taken into account. However, it is obvious that the merit of a win is commensurate to the strength of the beaten opponent. Next, we review existing player rating system.

3. THE ELO RATING SYSTEM

The performance of players cannot be measured absolutely. It can only be inferred from wins and losses. Therefore ratings have meaning only relative to other ratings. Both the average and the spread of ratings can be arbitrarily chosen. The ELO rating system is a method for calculating the relative strength of players. The ELO rating system [5] as developed by Arpad Elo is simple. Elo suggested estimating the true skill of players by updating their ratings when they won or lost against other players, based on a comparison with the other player's ratings. If a player won more games than he was expected to win, his rating would be adjusted upward, while if he won fewer games than expected his rating would be adjusted downward. For example, if the difference in ELO rating is 200 the stronger player is expected to win with a probability of 0.84. The formula to calculate a player's new rating based on his previous one is:

$$R_n = R_o + C * (S - S_e)$$

where R_n is the new rating, R_o is the old rating, S is the actual score (-1 for a loss, +1 for win), S_e is expected score, and C is learning rate constant.

4. PLAYER RATING INDUCTION

Without loss of generality, we will assume that no ties are possible (like in the game Hex). The entry $T_{i,j}$ of a tournament matrix T represents the number of wins of player i again player j. The rating of player k will be denoted by r_k.

We model the relationship between the strength of a player (win expectation) and its rating in a similar way to the ELO system. The probability $p_{i,j}$ that player i wins against player j is assumed to be $\text{logsig}(r_i - r_j)$ where

$$\text{logsig}(x) \stackrel{\text{def}}{=} \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{1+\exp(x)}$$

Notice that, as $\text{logsig}(x) + \text{logsig}(-x) = 1$, we have $p_{i,j} + p_{j,i} = 1$. Moreover,

$$\lim_{T_{i,j}+T_{j,i}\to\infty} \frac{T_{i,j}}{T_{i,j}+T_{j,i}} = p_{i,j}$$

Given a pair of players, we want to maximize the likelihood that the ratings of the players have generated the tournament results. In Section 5, we will first assume that the probabilities $p_{i,j}$ are known. This simplifying hypothesis will be relaxed in Section 6.

5. CASE WHERE THE $P_{I,J}$ ARE KNOWN

The equality $\text{logsig}(r_i - r_j) = p_{i,j}$ implies that

$$r_i - r_j = \log\left(\frac{p_{i,j}}{p_{j,i}}\right)$$

Therefore, to find ratings that agree with the win probabilities, we can minimize

$$\sum_{i,j}\left(r_i - r_j - \log\left(\frac{p_{i,j}}{p_{j,i}}\right)\right)^2$$

subject to the normalization constraint

$$\sum_k r_k = 0$$

The Lagragian of this constrained optimization problem is

$$L(r,\lambda) = \sum_{i,j}\left(r_i - r_j - \log\left(\frac{p_{i,j}}{p_{j,i}}\right)\right)^2 + \lambda\sum_k r_k$$

At the optimal point, we have $\frac{\partial L}{\partial r_k} = 0$ and $\frac{\partial L}{\partial \lambda} = 0$. The condition $\frac{\partial L}{\partial r_k} = 0$ yields

$$\sum_{j,k} 4 \left(r_k - r_j - \log\left(\frac{p_{k,j}}{p_{j,k}}\right)\right) + \lambda = 0$$

As $\sum_j r_j = 0$, this expression simplifies into

$$4\ n\ r_k - 4 \sum_j \log\left(\frac{p_{k,j}}{p_{j,k}}\right) + \ \lambda = 0$$

Where n is the number of players. Hence,

$$r_k = -\frac{\lambda}{4n} + \frac{1}{n}\sum_j \log\left(\frac{p_{k,j}}{p_{j,k}}\right) \quad (1)$$

Summing Equation (1) over k, gives

$$0 = \sum_k r_k = -\frac{\lambda}{4} + \frac{1}{n}\sum_{k,j} \log\left(\frac{p_{k,j}}{p_{j,k}}\right)$$

As $\sum_{k,j} \log\left(\frac{p_{k,j}}{p_{j,k}}\right) = 0$, we have $\lambda = 0$. The expression for r_k reduces to

$$r_k = \frac{1}{n}\sum_i \log\left(\frac{p_{k,i}}{p_{i,k}}\right) \quad (2)$$

Equation 2 shows that the larger the $p_{k,i}$, the larger the rating of player k. With this equation, the rating of player k can be interpreted as the average of $\log\left(\frac{p_{k,i}}{p_{i,k}}\right)$; its relative strength with respect to player i.

6. GENERAL CASE

Now, we assume that the only information available is the tournament matrix T. The probabilities $p_{i,j}$ are unknown. A principled approach to estimate the most likely values for the ratings, is to determine the values of the ratings that maximizes the likelihood of observing the tournament matrix T. That is, we want to find the $p_{i,j}$ that maximize

$$\prod_{i,j} \left(\begin{array}{c} T_{i,j} \\ T_{i,j} + T_{j,i} \end{array} \right) p_{i,j}^{T_{i,j}} \left(1 - p_{i,j}\right)^{T_{j,i}} \quad (3)$$

Taking the logarithm of the above expression, and dividing by $T_{i,j} + T_{j,i}$, it is easy to see that the maximization of (3) is equivalent to the maximization of

$$\sum_{i,j} \frac{T_{i,j}}{T_{i,j} + T_{j,i}} \log(p_{i,j}) + \frac{T_{j,i}}{T_{i,j} + T_{j,i}} \log(p_{j,i}) \quad (4)$$

The terms in Formula (4) are the opposite of the Kullback-Leibler distance between the Bernouilli probability distribution of parameter $\frac{T_{i,j}}{T_{i,j}+T_{j,i}}$ and the Bernouilli probability distribution of parameter $p_{i,j}$. It is well known that the terms in expression (4) are maximum when $p_{i,j} = \frac{T_{i,j}}{T_{i,j}+T_{j,i}}$.

Taking into account all pairs $\{i, j\}$, the expression that we wish to maximize is

$$\sum_{i,j} (T_{i,j} + T_{j,i}) \left(\frac{T_{i,j}}{T_{i,j}+T_{j,i}} \log(p_{i,j}) + \frac{T_{j,i}}{T_{i,j}+T_{j,i}} \log(p_{j,i}) \right)$$

This expression is the opposite of the cost function

$$\sum_{i,j} (T_{i,j} + T_{j,i})\ \text{dist}\left(\frac{T_{i,j}}{T_{i,j} + T_{j,i}}, p_{i,j}\right)$$

where $\text{dist}(p, q)$ represents Kullback-Leibler distance between the probability distribution p and q.

Unfortunately, there is no known close formula for the exact solution of this cost function. The trick we use is to generalize Formula 2 by considering that the factor $(T_{i,j} + T_{j,i})$ plays the role of an evidence weight. More importance should be given to pair of players that played a lot of games in the evaluation of the ratings.

To estimate the player ratings from the tournament matrix T, we propose to use the weighted formula

$$r_k = \alpha_k \sum_i (T_{k,i} + T_{i,k}) \log\left(\frac{T_{k,i}}{T_{i,k}}\right) \quad (5)$$

where α_k is the inverse of $\sum_i (T_{k,i} + T_{i,k})$.

In next section, we demonstrate experimentally that this weighted formula provides a more accurate estimate of the player ratings than the standard estimation and the uniform estimation (Formula 2).

7. EXPERIMENTAL RESULTS

To compare the rating estimation methods, we have randomly generated tournaments with the probability $p_{i,j}$ that player i wins against player j set to $\text{logsig}(r_i - r_j)$. The player ratings were generated with a normal distribution. The size of the population of players ranged from 10 to 80 and the average number of games per pair of players ranged from 2 to 32. To test the statistical significance of the comparisons we have used Wilcoxon signed rank test of equality of medians. The differences between the standard estimator and the weighted estimator are significant at a significance level of 0.05.

The boxplot figures (Figures 1,2 and 3) illustrate the difference between the rating estimators. The boxes have lines at the lower quartile, median, and upper quartile values. The whiskers are lines extending from each end of the boxes to show the extent of the rest of the data.

Even when the average number of games played between player pairs is low (around 2 or 3), the weighted estimator performs better than the standard estimator (see Figure 1). As the average number of games played between player pairs increases, the weighted estimator continues to improve its performance compared to the standard estimator (Figures 2, 3 and 4). As expected the uniform estimator and the weighted estimator behave similarly when the average number of games played between player pairs is high (above 10).

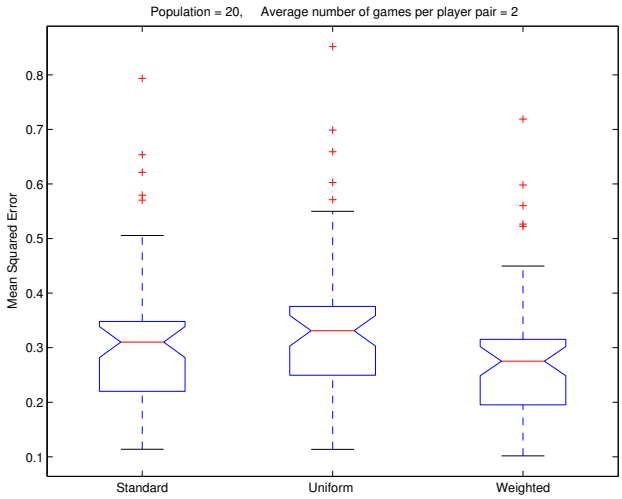

Figure 1: Mean errors are, 0.3195 for the standard estimator, 0.3395 for the uniform and 0.2870 for the weighted estimator.

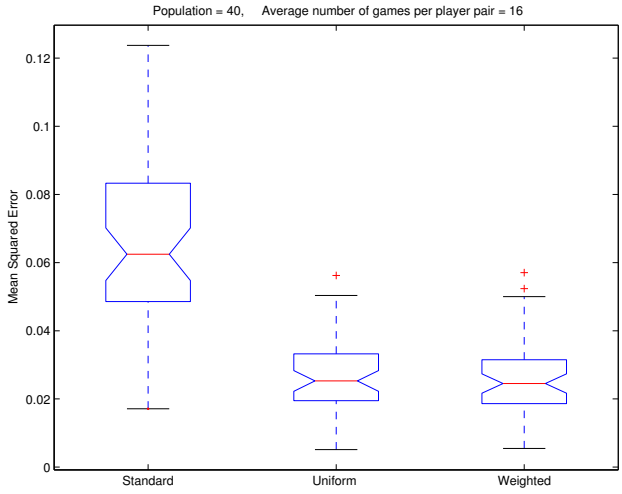

Figure 2: Mean errors are, 0.0687 for the standard estimator, 0.0269 for the uniform and 0.0260 for the weighted estimator.

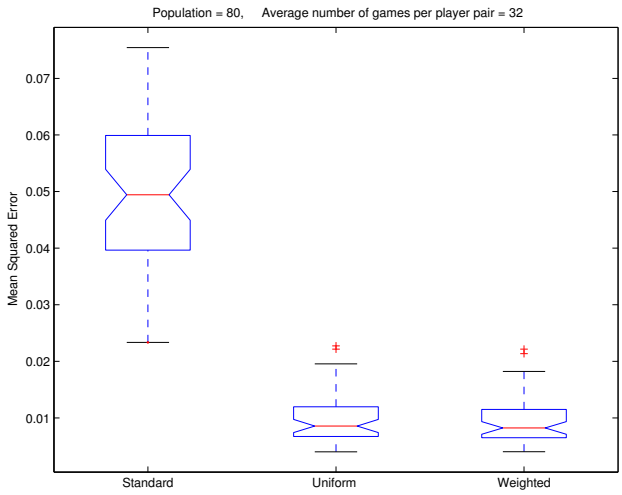

Figure 3: Mean errors are, 0.0494 for the standard estimator, 0.0098 for the uniform and 0.0097 for the weighted estimator.

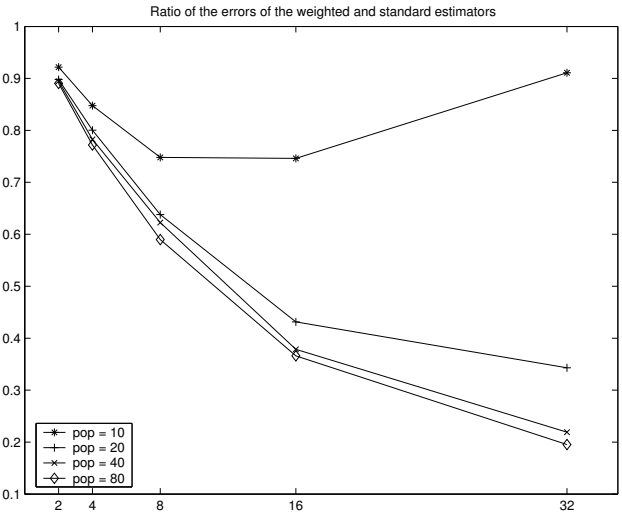

Figure 4: Ratios of the weighted estimator error over the standard estimator error. The horizontal axis corresponds to the average number of games per pair of players.

8. CONCLUSION

In this paper, we have proposed a method for evaluating player ratings from tournament results. With numerical experimental, we have demonstrated that the weighted estimator always provides a more accurate estimate than the standard estimator. For player populations of size larger than 10, the gain in accuracy is very significant. A more accurate estimation of the player ratings will allow a more informed selection process in artificial player evolution projects.

9. REFERENCES

[1] L. Barone and L. While. Adaptive learning for poker. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 566–573, 2000.

[2] J. Baxter, A. Tridgell, and L. Weaver. Knightcap: A chess program that learns by combining td(lambda) with gametree search. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 28–36, 1998.

[3] Alan Blair and Elizabeth Sklar. Exploring evolutionary learning in a simulated hockey environment. In Peter Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Congress on Evolutionary Computation*, 1999.

[4] K. Chellapilla and D. Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *Neural Networks, IEEE Transactions on*, 10(6):1382–1391, 1999.

[5] Arpad Elo. `http://www.thechessmill.com/html/ArpadElo.html`.

[6] Graham Kendall and Glenn Whitwell. An evolutionary approach for the tuning of a chess evaluation function using population dynamics. In *Congress on Evolutionary Computation*, 2001.

[7] T. Kojima, K. Ueda, and S. Nagano. An evolutionary algorithm extended by ecological analogy and its application to the game of go. In *Proceedings of the IJCAI97*, 1997.

[8] Jordan Pollack and Alan Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(3):225–240, 1998.

[9] A. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.*, 3(3), 1959.

[10] C. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41(4):256, 1950.

[11] Kenneth Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21, 2001. `http://www.jair.org`.

[12] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 1988.

[13] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6:215–219, 1994.

[14] S. Thrun. Learning to play the game of chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, 1995.